รีวิว
ประวิทย์ เรืองศาสตร์

สะกดรอย FoxPro 2.0

ข่าวคราวความคืบหน้าของโปรแกรมจัดการฐานข้อมูล FoxPro 2.0

เราอาจจะกล่าวได้ว่า โปรแกรม dBASE•II เป็นต้นตำรับของโปรแกรมจัดการฐานข้อมูล (DBMP : Database Management Package) บนเครื่องไมโครคอมพิวเตอร์เลยทีเดียว สาเหตุที่พอจะกล่าวอ้างได้ว่าดีเบสทูเป็นต้นตำรับก็คือ ความนิยมที่ไม่เสื่อมคลายที่มีต่อโปรแกรมตระกูลดีเบสนั่นเอง จนกระทั่งในปัจจุบันนี้ได้มีผู้ผลิตหลายรายที่ได้สร้าง DBMP ที่ใช้โครงสร้างของภาษาคล้ายกับโปรแกรมในตระกูลดีเบส (dBASE-like) ออกมาแข่งขันและต่อสู้กันในตลาดอย่างดุเดือด

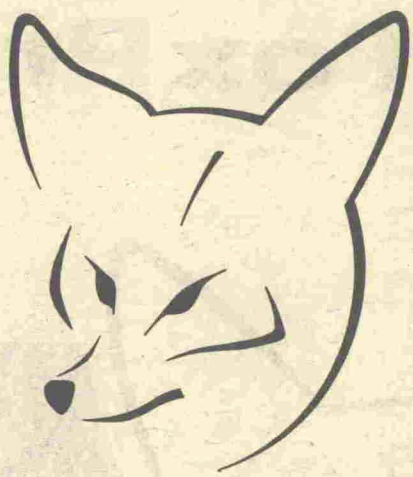
สำหรับผู้ใช้อย่างเรา ๆ ท่าน ๆ ที่นิยมชมชอบโปรแกรมประเภท dBASE-like ในตอนนี้ถือว่าเป็นช่วงเวลาที่คุ้มค่าสำหรับสิ่งที่เฝ้ารอเลยทีเดียว เพราะว่าโปรแกรมประเภทนี้ได้มีการพัฒนาขีดความสามารถขึ้นมาจากเดิมอย่างมาก มายเป็นต้นว่าบริษัท Nantucket ได้ออกตัวโปรแกรมคลิปเปอร์ เวอร์ชัน 5.0 ออกมาแล้ว โดยมีจุดเด่นที่การสร้างฟังก์ชันขึ้นเองโดยผู้ใช้ และเพิ่มส่วนของโอโอพี (OOP : Object Oriented Programming) ส่วนบริษัท Wordtech ก็ได้เข็นโปรแกรมที่มีชื่อว่า ARAGO ซึ่งเป็นโปรแกรมที่เกิดจากการรวมกันของโปรแกรม Quick silver และโปรแกรม dBXL โดยเป็นโปรแกรมที่มีความคอมแพตทิเบิลกับดีเบสโฟร์อย่างสมบูรณ์แบบ (dBASE IV Full Compatible) และยังเป็นคอมพายเลอร์ที่สามารถสร้างโปรแกรมแบบ .EXE ได้อีกด้วย ส่วนทางด้านบริษัท ฮัสตัน-เทจ ก็ได้มีโปรแกรมดีเบสโฟร์ที่ไม่ต้องโมษณาสรรพคุณ ซึ่งสามารถทำงานได้บนหลายระบบจัดการ เป็นต้นว่า Unix, MS-DOS, PC-DOS, VMS

ในตอนนี้บางท่านอาจจะมีคำถามอยู่ในใจว่า แล้วทางบริษัท ฟอกซ์ซอฟต์แวร์มัวไปทำอะไรอยู่ที่ไหน ใจเย็น ๆ ครับ เพราะว่าทางบริษัท ฟอกซ์

ซอฟต์แวร์ ผู้ซึ่งผลิตโปรแกรมฟอกซ์โปร ก็กำลังมุ่งทำงานอย่างหนักเพื่อที่จะสร้างฟอกซ์โปรเวอร์ชัน 2.0 ให้เป็น DBMP ที่ไร้เทียมทานอยู่เหมือนกัน เพียงแต่ว่าบริษัทฯ ไม่ได้โฆษณาถึงก้าวต่อไปของฟอกซ์โปรในลักษณะที่เป็นรูปเป็นร่างนัก เพราะหลังจากการเปิดเผยของฟอกซ์ ซอฟต์แวร์ เมื่อปลายปี 1989 เกี่ยวกับโครงการที่จะทำให้ฟอกซ์โปรเป็นคอมพายเลอร์ที่สร้างโปรแกรมแบบ .EXE ได้ ซึ่งถือได้ว่าเป็นส่วนที่ผู้ใช้รอคอยกันอยู่พอสมควร การโฆษณาถึงก้าวต่อไปของฟอกซ์โปรก็เงียบไปเลยทีเดียว

ที่มาของข่าวสาร

เมื่อเดือนมิถุนายนของปี 2533 ที่ผ่านมา ทางบริษัท ฟอกซ์ ซอฟต์แวร์ ได้เชิญบุคคลกลุ่มหนึ่งไปที่ เพอริสเบิก, มลรัฐโอไฮโอ (งานนี้ผู้เขียนไม่ได้รับเชิญนะครับ) เพื่อเข้าชมและแสดงความคิดเห็น ในส่วนที่เสร็จสมบูรณ์แล้ว (บางส่วน) ของโปรแกรม ฟอกซ์โปร เวอร์ชัน 2.0 โดยบุคคลกลุ่มนี้ได้ใช้เวลา 2 วัน ที่เพอริสเบิก จากนั้นด็อกเตอร์ P.L. Olympia (คนนี้แหละครับที่ได้รับเชิญไปในครั้งนั้น) ก็ได้เขียนบทความวิเคราะห์เรื่องราวต่าง ๆ เกี่ยวกับ ฟอกซ์โปร 2.0 จากข้อมูลที่เขาได้รับมาจากการไปในครั้งนั้น ซึ่งลงไว้ในวารสาร DBMS ฉบับที่ 12 ประจำเดือน พฤศจิกายน, 2533 ดังนั้นข้อมูลที่ท่านได้อ่านใน บทความอันนี้ (ซึ่งผู้เขียนเรียบเรียงจากบทความของ P.L. Olympia) ก็ต้องถือว่าเป็นข้อมูลเมื่อเดือนมิถุนายน, 2533 ดังนั้นเมื่อเวลาที่ฟอกซ์โปรออกจำหน่ายแล้วอาจจะมีบางส่วนที่แตกต่างไปจากสิ่งที่กล่าวไว้ในบทความอันนี้บางเล็กน้อย



(มีแนวโน้มในการเปลี่ยนแปลงไปในแนวทางที่ดีขึ้น) ในบทความอันนี้จะพิจารณาถึง ฟอกซ์โปร 2.0 ค่อนข้างละเอียดในแง่ของการทดสอบซอฟต์แวร์ก่อนออกจากวางตลาด (Bata Test Version) การนำเสนอครั้งนี้ก็เพื่อแจ้งข่าวความคืบหน้าของ ฟอกซ์โปรให้ผู้อ่านที่เป็นแฟนของฟอกซ์โปรได้อุ่นใจและยังเป็นประโยชน์แก่ผู้ที่พัฒนาระบบด้วยรองรับการเปลี่ยนแปลงที่จะเกิดขึ้นในอนาคตได้ด้วย

สิ่งที่ต้องคอย

กล่าวโดยย่อแล้ว สิ่งที่ทำให้พวกเราต้องคอยก็คือ **"สถาปัตยกรรมเปิด"** นั่นเอง เพราะว่าหลักการดังกล่าวนี้ จะส่งผลกระทบไปยัง ส่วนประกอบใหม่อื่น ๆ ทั้งหมด ยกตัวอย่างเช่น นักพัฒนาจะสามารถติดต่อกับฟอกซ์โปรโดยผ่านทาง External routine APIs (API : Application Program Interface) โดยความเห็นส่วนตัวแล้ว ผู้เขียนเห็นว่าเพียงส่วนนี้ส่วนเดียวก็คุ้มค่าคุ้มราคามากอยู่แล้ว และยังมีการเพิ่มคำสั่งและฟังก์ชันที่เปิดโอกาสให้จัดการกับหน้าต่าง, เมนู ของระบบได้ แล่ะสร้างโปรแกรมประยุกต์ ได้ใกล้เคียงกับรูปแบบที่พบในฟอกซ์โปรทุกอย่าง นอกจากนี้แล้วเพื่อเป็นการเปิดโอกาสให้นักพัฒนาสามารถจัดการกับส่วนประกอบต่าง ๆ ของโปรแกรมที่จะสร้างได้ง่ายขึ้น ฟอกซ์โปร 2.0 จึงได้เปลี่ยนแปลงในเรื่องของรูปแบบการเก็บข้อมูลของ report file, screen format file ซึ่งแต่ก่อนนั้นไฟล์พวกนี้จะจัดเก็บในรูปของ Binary format ซึ่งยากแก่การเข้าถึงและจัดการไม่สะดวก แต่ในฟอกซ์โปร 2.0 ไฟล์พวกนี้จะจัดเก็บในรูปของ .DBF ซึ่งจะสามารถเข้าถึงและจัดการได้ง่ายเหมือนกับแฟ้มข้อมูล .DBF ทั่ว ๆ ไป

คอมพายเลอร์

แต่ก่อนนั้นผลิตภัณฑ์ของฟอกซ์ซอฟต์แวร์ (FoxBase+, FoxPro 1.x) ก็เป็นคอมพายเลอร์อยู่แล้ว เพียงแต่เป็นคอมพายเลอร์เทียม (Pseudo Compiler) จึงได้มีคำถามมากมายที่ถามถึงว่าฟอกซ์โปร 2.0 จะเป็นคอมพายเลอร์แท้ (Conventional Compiler) ที่สร้างไฟล์แบบ .EXE หรือเปล่า? คำตอบก็คือ **ใช่และยังเป็นมากกว่านั้นอีก** แม้ว่าส่วนนี้จะยังไม่สมบูรณ์ ในตอนที่สาธิตเมื่อเดือนมิถุนายน เนื่องจากการสร้างคอมพายเลอร์เป็นเพียงแง่หนึ่งของ ส่วนอำนวยความสะดวกในการจัดการโครงการของฟอกซ์โปร (FoxPro's Project management facility) ซึ่งคล้าย ๆ กับโปรแกรม MAKE ของภาษามาตรฐาน (ยกตัวอย่างเช่นภาษา C, Pascal, Fortran) สำหรับรูปแบบของคอมพายเลอร์ของฟอกซ์โปรก็จะประกอบด้วย ไฟล์โครงการ (Project file) ซึ่งจะทำหน้าที่

เก็บข้อมูลที่เกี่ยวกับโปรแกรมที่จะสร้างขึ้น เป็นต้นว่าโปรแกรมไฟล์และแฟ้มข้อมูลที่ใช้ทั้งหมดรวมทั้งการติดต่อระหว่างไฟล์ เมื่อจะคอมพายล์โปรแกรมฟอกซ์โปรก็จะไปเอาข้อมูลต่าง ๆ จากโปรเจคไฟล์ โดยรูปแบบจริงแล้วโปรเจคไฟล์ก็คือแฟ้มข้อมูลอย่างหนึ่งรวมกับแฟ้มเมมโมไฟล์ (มีส่วนขยายเป็น .prj และ prt) เราสามารถสร้างโปรเจคไฟล์ขึ้นโดยการใช้ไดอะล็อคเมนู CREATE/MODIFLY PROJECT. หรืออาจใช้คำสั่งที่คอมมานด์ไลน์ (หรือสั่งไว้ในโปรแกรม) ได้โดยการใช้คำสั่งดังนี้

BUILD PROJECT (ชื่อโปรเจคไฟล์)

[FROM <ชื่อโปรแกรมหลัก> [,<ชื่อโปรแกรม>, ¦ <ชื่อไลบรารี่> [,..]]]

ฟอกซ์โปรจะสร้างโปรเจคไฟล์โดยเริ่มประมวลผลจากโปรแกรมและไลบรารี่ไฟล์ที่ระบุไว้ในคำสั่ง และทำการค้นหาโปรแกรมและไลบรารี่อื่น ๆ ที่อ้างถึง ที่นี้ก็อาจจะมีคำถามว่า ไลบรารี่ไฟล์ของฟอกซ์โปรคืออะไร? โดยฟอกซ์โปร APIs แล้ว ท่านสามารถเขียนฟังก์ชัน (คำสั่ง) ของท่านเองแล้วรวบรวมเอาไว้ในรูปแบบของรูทีนภายนอก จากนั้นก็จะใช้คำสั่ง SET LIB TO <ชื่อไลบรารี่> เพื่อให้ฟอกซ์โปรรู้จักกับฟังก์ชันเหล่านั้น โดยรูทีนภายนอกที่เขียนขึ้นเพื่อใช้กับฟอกซ์โปรอาจจะเขียนขึ้นด้วยภาษาแอสเซมบลีหรือภาษาซี แล้วทำการคอมไพล์ให้อยู่ในรูปแบบ .OBJ (ถ้าสงสัยเรื่องนี้ก็ต้องขอคิดเอาไว้ก่อนนะครับ แล้วผู้เขียนจะนำมาเขียนอีกครั้งในเรื่องของการใช้งานฟอกซ์โปร 2.0) จากนั้นก็นำเอาไฟล์ .OBJ มาสร้างไลบรารี่ไฟล์ด้วยการใช้คำสั่งดังนี้

BUILD LIB <ชื่อไลบรารี่> FROM <obj1>[,<obj2>...]]

หลังจากที่ท่านได้สร้างโปรเจคไฟล์เรียบร้อยแล้ว ท่านก็สามารถเลือกที่จะสร้างโปรแกรมได้สองแบบ คือ แบบเดิม (โปรแกรมที่ส่วนขยายเป็น .FXP) ซึ่งสร้างโดยคำสั่ง

BUILD APP <appfile> FROM <prjfile>

ซึ่งเวลาที่จะใช้งานก็ต้องมีฟอกซ์โปรเป็นตัวเรียกโปรแกรมของท่านอยู่ด้วยเสมอ (Runtime distribution) และอีกแบบคือ สร้างโปรแกรมที่สามารถรันได้ด้วยตัวเองโดยไม่ต้องอาศัยฟอกซ์โปร โดยโปรแกรมที่ได้จะมีส่วนขยายเป็น .EXE ด้วยคำสั่ง

BUILD EXE <exefile> FROM <prjfile>

สำหรับเรื่องรายละเอียดของรูทีนภายนอก ผู้เขียนจะไม่กล่าวในบทความอันนี้เพราะว่าเดี๋ยวจะเป็นเรื่องที่หนักเกินไป แต่โดยเค้าโครงที่ทราบมา รูทีนภายนอกสำหรับฟอกซ์โปรนั้นมีความสามารถในการติดต่อกับฟอกซ์โปรได้อย่างใกล้ชิดสนิทสนมพอสมควรทีเดียว ซึ่งก็พอจะมองเห็นได้จากการที่รูทีนภายนอกสามารถที่จะรับ-ส่งพารามิเตอร์กับฟอกซ์โปร 2.0 ได้เป็นอย่างดี และรูทีนภายนอกยังสามารถทำงานได้ในลักษณะแทรกระหว่างกระบวนการของฟอกซ์โปรได้อีกด้วย

ความเร็ว

สำหรับผู้ใช้ข้อมูลในฐานข้อมูลทุก ๆ ชีวิตแล้ว ความใฝ่ฝันอันยิ่งใหญ่ก็คือเรื่องของ **ความเร็ว** และจุดนี้โปรแกรมของทางฟอกซ์ซอฟต์แวร์ก็ได้ตอบสนองผู้ใช้มาตั้งแต่ต้นตามสโลแกนที่ว่า 'Nothing Runs like The Fox' บวกกับผลงานที่ผ่านมาไม่ว่าจะเป็นฟอกซ์เบส-พลัส, ฟอกซ์โปร เวอร์ชัน 1.x ก็เป็นการรับรองคุณภาพได้อยู่แล้ว เพราะในการทดสอบ Benchmark โดยทั่ว ๆ ไป ผลิตภัณฑ์ของฟอกซ์ฯ ก็ชนะคู่แข่งขันในเรื่องความเร็วอยู่แล้ว ดังนั้นจึงสามารถบอกได้เลยว่าฟอกซ์โปร 2.0 ต้องเร็วแน่นอน แต่ท่านอาจจะมีคำถามอยู่ว่าเร็วอย่างไร? แค่ไหน? ก็ต้องมาดูผลการทดสอบกันล่ะครับ จากการทดสอบโดยใช้เครื่อง 386 แล็ปทอป ที่มีหน่วยความจำถึง 10 เมกะไบต์ ทำการสอบถามข้อมูลที่มีความซับซ้อนในแฟ้มข้อมูลขนาด 30 เมกะไบต์ ปรากฏว่าฟอกซ์โปร 2.0 ใช้เวลาในการสอบถามเพียง 0.5 วินาที เอาล่ะบางท่านอาจจะเห็นว่าเป็นเพราะว่าเครื่องที่ใช้ทดสอบมันเร็วบวกกับมีหน่วยความจำมากนะซิถึงได้เร็ว ก็เลยมีการทดสอบโดยการจำลองเครื่อง 286 โดยปิดส่วนที่เป็นหน่วยความจำเพิ่มขยาย (EMS : Expanded Memory Specification) เอาไว้และทำการลดความเร็วของซีพียูลงอีกแล้วทำการสอบถามอย่างเดิมปรากฏว่าใช้เวลาเพียง 1.5

วินาที ถึงแม้ว่าการทดสอบจะไม่สมบูรณ์นัก เพราะว่าต้องใช้เครื่อง 286 จำลอง (เหตุที่ต้องจำลอง เพราะว่าห้างห้องปฏิบัติการไม่มีเครื่อง 286 ที่มีหน่วยความจำ 640 กิโลไบต์อยู่เลย) แต่ก็ทำให้เห็นแนวทางในเรื่องของความเร็วว่าเราจะคาดได้แค่ไหนจากฟอกซ์โปร 2.0 เพราะว่าฟอกซ์โปรได้ใช้เทคโนโลยีของการออฟติไมต์แบบใหม่ เพื่อให้ประสิทธิภาพสูงขึ้น ซึ่งกล่าวอ้างไว้ว่าใกล้เคียงกับซอฟต์แวร์บนเครื่องเมนเฟรม จากผลการทดสอบอันนี้ทำให้ผู้เข้าสังเกตการณ์ท่านหนึ่ง ได้เล่าถึงเหตุการณ์ในสมัยก่อน เกี่ยวกับเรื่องของความเร็วว่า แต่ก่อนนี้ได้มีผู้ขายไมโครคอมพิวเตอร์ (ของเมืองนอก) ที่ไร้คุณธรรมได้สาธิตเครื่องไมโครคอมพิวเตอร์ของตนโดยใช้เครื่องมินิคอมพิวเตอร์ (VAX) เพื่อให้เกิดผลลัพธ์ที่น่าพึงพอใจสำหรับซอฟต์แวร์ที่จะใช้ (สำหรับเมืองไทยก็ไม่น่าจะมีผู้ขายประเภทนี้อยู่นะครับ) แต่ต่อไปผู้ขายมินิคอมพิวเตอร์ อาจจะสาธิตความสามารถของเครื่องของตนโดยใช้ฟอกซ์โปรร่วมกับเครื่องแล็ปท็อปก็เป็นไปได้

ส่วนขยายจากดีเบสโฟร์

ส่วนหนึ่งของ dBASE IV ที่ผู้ใช้รู้สึกชอบอกชอบใจมากก็คือ เรื่องของการสร้างดัชนีให้แก่แฟ้มข้อมูล โดยสามารถรวมคีย์ถึง 47 คีย์ ไว้ในแฟ้มดัชนีเพียงแฟ้มเดียว (Multiple Index facility : จะสร้างแฟ้มดัชนีที่มีส่วนขยายเป็น .MDX) ซึ่งเป็นจุดที่ดีเบสโฟร์สามารถแก้ปัญหาเรื่องการบำรุงรักษาแฟ้มดัชนีลงไปได้ เพราะแต่เดิมนั้นเมื่อมีแฟ้มดัชนีมาก ๆ ก็จะเกิดปัญหาความไม่ถูกต้องของแฟ้มดัชนี เพราะในบางครั้งโปรแกรมอาจจะไม่ได้เปิดแฟ้มดัชนีทั้งหมดออกมาใช้ ดังนั้นการปรับปรุงค่าพอยน์เตอร์ในแฟ้มดัชนีจึงมีโอกาสผิดพลาดขึ้นได้แต่วิธีการของ Multiple Index นั้นจะทำการปรับปรุงค่าคีย์ต่าง ๆ ในแฟ้มดัชนีโดยอัตโนมัติ ซึ่งแต่ก่อนนี้ถือว่าดีเบสโฟร์เด่นมากในเรื่องนี้เมื่อเทียบกับโปรแกรมประเภทเดียวกัน ดังนั้นในฟอกซ์โปร 2.0 จึงได้มีวิธีการนี้เช่นกัน โดยใช้ชื่อวิธีการนี้ว่า Compound Index facility (สร้างแฟ้มดัชนีที่มีส่วนขยายเป็น .CDX) โดยทำหน้าที่คล้ายกันกับแฟ้มดัชนี .MDX ของดีเบสโฟร์ แต่สำหรับแฟ้มดัชนี .CDX ในฟอกซ์โปรมีข้อดีที่เพิ่มขึ้นมาอีกก็คือ สามารถสร้างคีย์ขึ้นในแฟ้ม .CDX ได้โดยไม่จำกัดจำนวน สำหรับจุดเด่นอีกอย่างหนึ่งของแฟ้มดัชนี .CDX ก็คือ สำหรับทุกคีย์ดัชนีสามารถที่จะใช้คำสั่ง SET ORDER TO <field or index> DESCENDING เพื่อเป็นการระบุให้ข้อมูลเรียงจากมากไปหาน้อยสำหรับคีย์นั้น ๆ เมื่อเทียบกับดีเบสโฟร์แล้ว ถ้าหากต้องการจัดเรียงข้อมูลในคีย์ดัชนีใด ๆ ให้อยู่ในรูป DESCENDING ORDER ก็ต้องทำการสร้างคีย์ดัชนีเพิ่มขึ้นอีก โดยระบุในตอนสร้างคีย์ด้วยคำว่า DESCENDING ซึ่งก็หมายถึงการที่ท่านจะต้องสิ้นเปลืองเนื้อที่ของอุปกรณ์เก็บข้อมูลของท่านไปอีกนั่นเอง

ในฟอกซ์โปรจะสร้างแฟ้มดัชนีได้สองชนิดคือ REGULAR และชนิด COMPACT สำหรับแฟ้มดัชนีชนิด REGULAR จะคอมแพตกับแฟ้มดัชนีของ FoxBase+ ส่วนแฟ้มดัชนีชนิด COMPACT จะไม่คอมแพตกับแฟ้มดัชนีของ FoxBase+ แต่มีข้อดีคือกินเนื้อที่ดิสค์น้อยลง คือใช้เนื้อที่เพียงหนึ่งส่วนหกของแฟ้มดัชนีชนิด REGULAR สำหรับการสร้างแฟ้มดัชนีชนิด COMPACT สามารถสร้างได้โดยใช้คำสั่ง INDEX ON <ndx_expsn> COMPACT ฟอกซ์โปรก็จะสร้างแฟ้มดัชนีชนิด COMPACT ขึ้น แต่รายละเอียดในการใช้งานแล้ว ยังไม่มีข้อมูลแจ้งว่าระหว่างแฟ้มดัชนี 2 ชนิดนี้ให้ผลเรื่องประสิทธิภาพแตกต่างกันอย่างไร? แค่ไหน?

สำหรับคำสั่งในดีเบสโฟร์ที่ผู้ใช้ฟอกซ์โปรขาดหายไป (และต้องการ) ก็คือคำสั่ง SET SKIP TO ที่ใช้ร่วมกับคำสั่ง SET RELATION ซึ่งใช้ในการเชื่อมโยงความสัมพันธ์ระหว่างแฟ้มข้อมูลแบบหนึ่งต่อหนึ่งต่อมากกว่า (one to many) และมากกว่าต่อมากกว่า (many to many) ซึ่งจะช่วยทำให้การเข้าถึงข้อมูลระหว่างแฟ้มหลักกับแฟ้มรายละเอียดทำได้โดยสะดวก สำหรับในฟอกซ์โปร 2.0 ก็ได้พัฒนาคำสั่ง SET SKIP TO ขึ้นมาเพื่อตอบสนองความต้องการของผู้ใช้ด้วย

เนื่องจากดีเบสโฟร์ได้มีส่วนของ SQL (Structure Query Language)

เพิ่มเข้ามาทำให้เด่นขึ้น แต่ทว่าผู้ใช้ส่วนใหญ่ยังเห็นว่า SQL ในดีเบสโฟร์ยังช้าเหมือนกับเต่า นอกจากนี้แล้วการใช้งานก็ยังไม่สะดวกนัก เพราะเวลาที่ต้องการใช้งานก็ต้องใช้คำสั่ง SET SQL ON เสียก่อน จากนั้นดีเบสโฟร์ก็จะแปลงคำสั่ง SQL ให้เป็นคำสั่งดีเบสเสียก่อนแล้วจึงสามารถรันได้ แต่สำหรับฟอกซ์โปร 2.0 แล้วท่านสามารถใช้คำสั่ง SQL ได้ทันทีโดยไม่ต้องใช้คำสั่ง SET SQL ON โดยท่านสามารถพูดจาภาษา SQL ได้ทันทีและยังพูดได้อย่างไม่เพี้ยนอีกด้วย ยกตัวอย่างเช่นในการเขียนรูปแบบของคำสั่งของภาษา SQL (SELECT <whatever> WHERE <condition>) เราสามารถใช้คำสั่ง AND, OR ได้โดยไม่ต้องใส่จุดครอบไว้เหมือนก่อน (แต่ก่อนนั้นการใช้ AND,OR เป็นดังนี้ .AND., .OR.) ดังนั้นเงื่อนไขที่เขียนไว้หลัง WHERE ในฟอกซ์โปร 2.0 ก็จะมีรูปแบบเหมือนกับภาษา SQL จริง ๆ (แต่ถ้าหากต้องการเขียนแบบเดิมก็ได้) ภาษา SQL ในฟอกซ์โปรจึงเป็น SQL มาตรฐาน (ANSI SQL) และยังมีส่วนขยายอีกด้วย นอกจากนี้แล้ว ยังมีส่วนของ RQBE (Relational Query By Example) ซึ่งเป็นรูปแบบพิเศษของ QBE อีกด้วย

เครื่องมือช่วยสร้างโปรแกรม

โดยหลักการของสถาปัตยกรรมเปิดนี่เอง ที่ทำให้ฟอกซ์ซอฟต์แวร์ได้จัดเตรียมเครื่องไม้เครื่องมือในการสร้างโปรแกรม ให้สามารถสร้างรูปแบบการติดต่อกับผู้ใช้ ได้เหมือนกับที่ปรากฏในฟอกซ์โปรเอง เครื่องมือที่ช่วยในการออกแบบและสร้างที่ได้จัดเตรียมไว้ก็คือ Screen painter ซึ่งจะช่วยให้นักพัฒนา

สามารถปรับแต่งและสร้างองค์ประกอบต่าง ๆ บนจอภาพได้หลายองค์ประกอบ (สำหรับองค์ประกอบของจอภาพที่สามารถสร้างขึ้นโดยฟอกซ์โปรได้ให้คำอธิบายไว้ในรูปที่ 1)

การเรียกใช้โปรแกรม Screen painter สามารถใช้คำสั่ง CREATE/MODIFY SCREEN ข้อมูลทั้งหมดเกี่ยวกับองค์ประกอบของจอภาพที่สร้างขึ้นจะถูกบันทึกไว้ใน แฟ้มข้อมูล/แฟ้มเมมโม ที่มีส่วนขยายเป็น .SCX และ .SCT ซึ่งท่านสามารถจะจัดการกับไฟล์ทั้งสองนี้ได้เหมือนกับแฟ้มข้อมูล .DBF และแฟ้มเมมโม. FPT ทั่ว ๆ ไป

ในส่วนนี้ ฟอกซ์โปร 2.0 ยังประกอบด้วยโปรแกรมสร้างระบบงานประยุกต์ (Application Builder) ที่สามารถนำเอาข้อมูลที่สร้างขึ้นจากโปรแกรม Screen painter มาทำการสร้างโปรแกรม ซึ่งจะให้ผลลัพธ์ออกมาได้หลายรูปแบบ เป็นต้นว่า ฟอร์แมตไฟล์, โปรแกรมโมดูล หรือเป็นระบบงานที่เสร็จสมบูรณ์

ภาษาสร้างแบบ

ในฟอกซ์เบสพลัส 2.1 และฟอกซ์โปร 1.x ก็มีภาษาสร้างแบบ (Template Language) และโปรแกรมสร้างโปรแกรมอยู่ก่อนแล้ว ซึ่งประกอบด้วยโปรแกรม Foxview, FoxCode, FoxGen แต่ทว่าการที่ท่านจะใช้ภาษาสร้างแบบชุดนี้ได้ท่านต้องเรียนรู้ไวยกรณ์ของภาษาอันนี้ให้ดีเสียก่อน เพราะว่าเป็นภาษาที่ต่างไปจากภาษาของฟอกซ์เบสพลัสและฟอกซ์โปร 1.x (โดยมีคู่มือแยกต่างหากอีก 138 หน้า และเป็นไวยกรณ์ที่ท่านอาจจะไม่คุ้นเคยนัก) แต่สำหรับฟอกซ์โปร 2.0 แล้ว ท่านไม่จำเป็นต้องจำไวยกรณ์ของภาษาสร้างแบบอีกเลย นอกจากจำคำสั่งเพียงเล็กน้อยที่แสดงไว้ในรูปที่ 2 เท่านั้น ดังนั้น

Controls : คือการกำหนดว่าต้องการให้รับข้อมูล/คำสั่งเข้าสู่โปรแกรมในรูปแบบใด ซึ่งประกอบด้วย Check boxes, menu popups, scrollable lists, text buttons, radio buttons
Fields : หมายถึงฟิลด์ของแฟ้มข้อมูล, ค่าของตัวแปร, ค่าที่ได้จากฟังก์ชันที่กำหนดขึ้นโดยผู้ใช้
Graphics : หมายถึงเส้นและกรอบ
Text : หมายถึงข้อความตัวอักษร

รูปที่ 1 องค์ประกอบที่สร้างขึ้นด้วย Screen painter

```
- SET TEXTMERGE ON | OFF | DELIMITERS [<expC1>, <expC2>]] [TO
  <file> [ADDITIVE]] [WINDOW <win_name>]
- TEXT [TO <expN>] and ENDTEXT
- \ <textline> and \ \ <textline>
- _PRETEXT = <expC>
- TEXT = <expN>
** หมายเหตุ : สำหรับรายละเอียดการใช้งานจะนำมากล่าวในบทความเรื่อง
การใช้งานฟอกซ์โปร 2.0 อีกครั้งในเร็ว ๆ นี้
```

รูปที่ 2 ตัวอย่างภาษาสร้างแบบ

ในฟอกซ์โปร 2.0 เรื่องจู้จี้น่ารำคาญเกี่ยวกับภาษาสร้างแบบจึงหมดไปโดยสิ้นเชิง

READs/GETs ที่ดีขึ้น

เกี่ยวกับส่วนที่น่าสนใจของคำสั่ง @ GET..., READ ได้แก่คำสั่ง CLEAR READ ที่เพิ่มขึ้นมา ซึ่งทำหน้าที่หยุดและออกจากการทำงานของคำ READ ที่กำลังปฏิบัติการอยู่ คำสั่ง READ ในฟอกซ์โปร สามารถใช้ซ้อนกันได้ถึง 4 ระดับและสามารถเช็คได้ว่าตอนนี้กำลังอยู่ในระดับใด โดยใช้ฟังก์ชัน RDLEVEL () และคำสั่ง READ ยังมีพารามิเตอร์ให้เลือกใช้ได้สองแบบคือ

- CYCLE เป็นการบอกให้ทำการรับข้อมูลจาก Get fields โดยไม่หยุดการทำงานที่ฟิลด์สุดท้าย แต่จะวนกลับไปยังฟิลด์แรกอีก จนกว่าจะมีการกดคีย์ Ctrl-W หรือ Esc

- OBJECT เป็นการสร้างองค์ประกอบต่าง ๆ ขึ้นระหว่างการ READ (ซึ่งประกอบด้วยองค์ประกอบที่แสดงและให้ความหมายไว้ในรูปที่ 1) สำหรับการใช้ก็จะเป็นการใช้ในรูปของคำสั่ง PICTURE และ FUNCTION ในคำสั่ง @...GET และยังใช้ร่วมกับคีย์เวิร์ด ENABLE/DISABLE อีกด้วย (จุดนี้ผู้เขียนจะไม่เน้นเรื่องลักษณะของรูปแบบมากนัก เพราะอาจจะมีการเปลี่ยนแปลงภายหลังได้) ในเรื่องของคำสั่ง GET/READ ยังมีคำสั่ง SHOW GET ที่ใช้ในการแสดงค่าในฟิลด์ (รวมทั้งองค์ประกอบอื่น ๆ ซ้ำภายหลังคำสั่ง READ อีกด้วย)

นอกจากนี้ในฟอกซ์โปร 2.0 ท่านยังสามารถสร้างย่านการรับข้อมูลจากตัวแปร, ฟิลด์ข้อมูลในรูปแบบของการแก้ไขฟิลด์เมมโม ซึ่งจะสามารถใช้ความสามารถของ Block Operation, Word Wrap ซึ่งจะปรากฏในรูปแบบของ

คำสั่ง @...EDIT ซึ่งมีลักษณะการใช้ใกล้เคียงกับ @...GET

เกี่ยวกับเมนู

ในฟอกซ์โปร 2.0 ผู้ใช้มีความสามารถในการควบเมนูของระบบได้อย่างใกล้เคียงกับเมนูที่สร้างขึ้นเอง โดยส่วนประกอบของเมนูของระบบแต่ละส่วนก็จะมีชื่อเฉพาะให้อ้างถึงได้ เป็นต้นว่า ถ้าต้องการที่จะลบ Window pad ของเมนูระบบและลบ Filer bar ของ System pad ก็สามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

RELEASE PAD _msm_windo OF _msysmenu

RELEASE BAR _mst_filer OF _msym_systm

สำหรับคำสั่งของการสร้างเมนู ซึ่งได้แก่กลุ่มคำสั่ง DEFINE BAR, DEFINE PAD, DEFINE MENU, DEFINE POPUP ก็ได้เพิ่มพารามิเตอร์ของคำสั่งขึ้นมาอีกคือ KEY และ MARK พารามิเตอร์ โดยพารามิเตอร์ KEY ใช้ในการกำหนดคีย์ฉับไว (Hot key) สำหรับเลือกรายการเมนู (ซึ่งอาจจะกำหนดให้เป็นคีย์เดี่ยวหรือให้กดหลายคีย์พร้อมกัน (combination key ยกตัวอย่างเช่น Alt+X) และพารามิเตอร์ MARK ใช้สำหรับการกำหนดตัวอักษร ที่บ่งบอกให้ผู้ใช้กดเพื่อเลือกรายการเมนู สำหรับคำสั่ง DEFINE BAR/PAD ยังมีพารามิเตอร์เพิ่มเติมคือ BEFORE/AFTER ที่ใช้ในการกำหนดตำแหน่งและอันดับในการแสดงผลของ bar หรือ pad คำสั่ง DEFINE POPUP ยังสามารถกำหนด TITLE และ FOOTER เพื่อแสดงข้อความที่กรอบบนหรือกรอบล่างของเมนูแบบ

```
MOVE POPUP  : ใช้ในการเคลื่อนย้าย POPUP MENU ไปตำแหน่งใหม่
SIZE POPUP  : ใช้เปลี่ยนแปลงขนาดของ POPUP
ON BAR      : ใช้กระตุ้นให้มีการทำงานตามรายการที่ถูกเลือก
ON SELECTION BAR : กำหนดการกระทำ (รูทีน) ให้กับ รายการ POPUP
ON SELECTION MENU : กำหนดการกระทำ (รูทีน) ให้กับรายการ MENU BAR
SET MARK OF : เป็นการกำหนด Check mark Character ให้กับรายการ
              POPUP หรือ รายการของ PAD MENU
```

รูปที่ 3 คำสั่งใหม่ ๆ ที่เกี่ยวกับเมนู

```
CNTBAR 0  : ให้ค่าจำนวนของรายการในพอบอับเมนู
CNTPAD 0  : ให้ค่าจำนวนของ pad ใน pad menu
GETBAR 0  : ให้ค่าเป็นชื่อของ bar ที่ระบุในพอบอับเมนู
GETPAD 0  : ให้ค่าเป็นชื่อของ pad ที่ระบุใน pad menu
MRKBAR 0  : ให้ค่าเป็น .T. ถ้าหาก bar ที่ระบุถูก Mark
MAKPAD 0  : ให้ค่าเป็น .T. ถ้าหาก pad ที่ระบุถูก Mark
            (ดูคำสั่ง SET MARK OF ในรูปที่ 3)
PRMBAR 0  : ให้ค่าข้อความของรายการที่ระบุ
PRMPAD 0  : ให้ค่าข้อความของ pad ที่ระบุ
```

รูปที่ 4 ฟังก์ชันใหม่ ๆ ที่เกี่ยวกับเมนู

POPUP สำหรับคำสั่งเกี่ยวกับเมนูที่เพิ่มขึ้นได้แสดงไว้ใน รูปที่ 3 และ 4

FoxReport มีอะไรใหม่

สิ่งแรกที่จะต้องกล่าวถึงก็คือ ไฟล์ .FRX ที่เก็บรายละเอียดต่าง ๆ ของรูปแบบรายงานที่สร้างขึ้น ในฟอกซ์โปร 2.0 ไฟล์ ท่านสามารถจัดการกับ .FRX ได้เหมือนกับเป็นแฟ้มข้อมูล (.DBF) ทั่ว ๆ ไป ดังนั้นจึงเป็นเรื่องง่ายที่จะแก้ไขเปลี่ยนแปลงรูปแบบรายงานที่ได้ออกแบบไว้ (ซึ่งอาจจะเป็นการแก้ไขหรือเพียงเล็กน้อย) ได้โดยไม่ต้องเข้าสู่โปรแกรม FoxReport

สำหรับการสร้างรูปแบบรายงานก็ได้เพิ่มความสามารถขึ้นอีก นั่นคือท่านสามารถเปิดหน้าต่างสำหรับออกแบบ (Layout Window) ได้มากกว่าเดิมซึ่งเปิดได้เพียง 1 วินโดว์ (จำนวนหน้าต่างที่เปิดได้ขึ้นอยู่กับขนาดของหน่วยความจำ) และในหน้าต่างออกแบบ ท่านสามารถที่จะกำหนดตำแหน่งขององค์ประกอบต่าง ๆ (ข้อความ, วันที่, ฟิลด์ข้อมูล) ได้อย่างสะดวก และถ้าหากองค์ประกอบที่ท่านกำหนดขึ้นเกินขนาดของหน้าต่าง หน้าต่างก็จะขยายขนาดให้โดยอัตโนมัติ สำหรับทางเลือกในการจัดข้อมูลก็มีให้เลือก 2 แบบคือ Group: จัดเรียงข้อมูลแยกเป็นกลุ่มโดยใช้ฟิลด์ที่กำหนด และ Ungroup: จะจัดเรียงข้อมูลตามลำดับเดิมในแฟ้มข้อมูล

นอกจากนี้แล้วท่านยังสามารถสร้างฟิลด์คำนวณได้อีก โดยมีฟังก์ชันการคำนวณให้เลือกใช้เช่น Sum, Average, Count ตลอดจนฟังก์ชันทางสถิติ เช่น ค่าส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation Value), ค่าความแปรปรวน (Variance Value)

การนำเข้า-ออกข้อมูล

เรื่องการนำเข้า-ส่งออก ข้อมูลกับโปรแกรมอื่น ๆ บางทีอาจไม่ใช่เรื่องจำเป็นมากนัก แต่ก็จำเป็นเหมือนกัน ขอยกตัวอย่างเฉพาะการส่งออกข้อมูล เช่นงานบางอย่างฟอกซ์โปรยังไม่มีส่วนที่สนับสนุน เป็นต้นว่า การเขียนกราฟ (ซึ่งส่วนนี้มักมีในซอฟต์แวร์สเปรดชีต อันได้แก่ โปรแกรมโลตัส 1-2-3, ไมโครซอฟต์เอ็กซ์เซล ฯลฯ) เราก็ต้องส่งข้อมูลที่ต้องการเขียนกราฟ ออกไปให้โปรแกรมเหล่านั้นจัดการให้ ซึ่งแต่ก่อนนี้อาจจะเป็นเรื่องยุ่งยากพอสมควรทีเดียว แต่ทว่าใน 2.0 ได้อำนวยความสะดวกให้สามารถเขียนข้อมูลไปยังไฟล์ของโปรแกรมต่าง ๆ ได้โดยระบุชนิดของไฟล์ที่ต้องการส่งออก และสามารถอ่านข้อมูลจากไฟล์ของโปรแกรมอื่น ๆ เข้ามาเก็บยังแฟ้มข้อมูลของฟอกซ์โปรได้โดยตรง โดยชนิดของไฟล์ที่ฟอกซ์-

โปรสามารถรับเข้าและส่งออกได้โดยตรง ได้แสดงไว้ในรูปที่ 5 โดยรูปแบบการใช้จะเป็นดังนี้

การนำออก

USE <source_file> ←แฟ้มข้อมูลที่ต้องการนำข้อมูลออก

COPY TO <destination_file> TYPE <ชนิดของไฟล์>

การนำเข้า

USE <destination_file> ←แฟ้มข้อมูลที่ต้องการนำเอาข้อมูลเข้า

APPEND FROM <source_file> TYPE <ชนิดของไฟล์>

จะเห็นว่าคำสั่งทั้งสองนี้ทำให้เราสามารถใช้ข้อมูลร่วมกับโปรแกรมอื่น ๆ ได้ไม่ยุ่งยากเลย นอกจากนี้แล้วในฟอกซ์โปร 2.0 ยังมีคำสั่งใหม่คือ IMPORT, EXPORT ที่ทำงานได้กับไฟล์ตามรูปที่ 5 และอำนวยความสะดวกในการแลกเปลี่ยนข้อมูลกับ DBMP และโปรแกรมสเปรดชีตอื่น ๆ ได้อีก

เรื่องของอะเรย์

ในเรื่องของอะเรย์แล้วดูเหมือนว่าคลิปเปอร์จะเหนือกว่าโปรแกรมประเภทเดียวกันอยู่ในเรื่องของฟังก์ชันที่จัดการเกี่ยวกับอะเรย์ เป็นต้นว่าในฟอกซ์โปร 1.0 ถ้าหากต้องการที่จะรู้ถึงขนาดหรือมิติของอะเรย์ ผู้ใช้ก็ต้องอาศัยการสร้างฟังก์ชันขึ้นใช้เอง (User-defined Function) ดังที่ได้แสดงไว้ในรูปที่ 6 ดังนั้นผู้ใช้ก็เลยอยากจะให้ฟอกซ์โปรสนับสนุนเรื่องเกี่ยวกับอะเรย์ให้ใกล้เคียงกับคลิปเปอร์ ดังนั้นในฟอกซ์โปร 2.0 จึงได้มีฟังก์ชัน ASIZE () เพื่อให้ค่า จำนวนสมาชิก, จำนวนแถว, จำนวนคอลัมน์ในอะเรย์ แต่ทว่าเท่านี้ก็ยังไม่พอ และจะเป็นเรื่องประหลาดมากถ้าหากฟอกซ์โปร 2.0 (ตอนที่เสร็จสมบูรณ์แล้ว) ไม่ได้เพิ่มฟังก์ชันจัดการเรื่องอะเรย์ขึ้นไปอีก เพราะเสียงเรียกร้องของผู้ใช้ต้องการความสามารถด้านอะเรย์ที่ใกล้เคียงกับโปรแกรมคลิปเปอร์มีมากมายเหลือเกิน

ชนิดของไฟล์	ใช้ในโปรแกรม
DIF	รูปแบบแฟ้มที่ใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูล
FW2	Framework II (Spreadsheet)
MOD	Microsoft Multiplan V.4.x (Binary file)
RPD	Rapidfile databases (Database)
SYLK	Multiplan (Symbolic Link Interchange)
WK1	Lotus 1-2-3 Release 2.x (Spreadsheet)
WK3	Lotus 1-2-3 Release 3.0 (Spreadsheet)
WKS	Lotus 1-2-3 Release 1A (Spreadsheet)
WR1	Symphony ver. 1.1, 1.2, 2.0 (Spreadsheet)
WRK	Symphony ver. 1.0, 1.01 (Spreadsheet)
XLS	Microsoft Excel version 2.0 (Spreadsheet)

รูปที่ 5 ชนิดของไฟล์ข้อมูลที่เปลี่ยนได้

```
Function A_LEN
PARAMETERS Arr
FOR i = 1 TO 3600 && Max ของขนาดอะเรย์
   IF TYPE ("&Arr[i]") = "U"
      RETURN i-1
   ENDIF
ENDFOR
RETURN 3600
```

รูปที่ 6 การสร้างฟังก์ชันขึ้นใช้เอง

เล็ก ๆ น้อย ๆ เกี่ยวกับ Window

ท่านที่เคยใช้ฟอกซ์โปร 1.x มาแล้วก็คงจะสนุกกับการใช้วินโดว์โดยการใช้คำสั่ง DEFINE WINDOW กันพอสมควร และในฟอกซ์โปร 2.0 เรื่องของวินโดว์ก็ได้เพิ่มพารามิเตอร์ที่ใช้ระบุคุณลักษณะของวินโดว์เพิ่มขึ้นมาอีก (เพื่อเพิ่มความสะดวกและความสวยงาม) ซึ่งได้แก่พารามิเตอร์ดังต่อไปนี้

- FOOTER ใช้ในการกำหนดข้อความที่จะแสดงผลที่กรอบล่างของวินโดว์

- FILL ใช้ในการกำหนดอักษรที่จะใช้เป็นพื้น (Background) ของวินโดว์

- MINIMIZE เป็นการกำหนดขนาดที่เล็กที่สุดของวินโดว์เพื่อให้พื้นที่แก่วินโดว์ที่กำลังใช้งานอยู่ และยัง

สามารถปรับขนาดและตำแหน่งของวินโดว์ด้วยคำสั่ง ZOOM เพื่อเพิ่มหรือลดขนาดของวินโดว์ในฟอกซ์โปร 2.0 ยังมีฟังก์ชันที่เกี่ยวกับวินโดว์เพิ่มขึ้นอีก 2 ฟังก์ชันก็คือ

- WCHILD (<win_name>) : ให้ค่าเป็นของจำนวนหรือชื่อ Child's window ออกมา

- WPARENT (<win_name>) : ให้ค่าเป็นชื่อของ Parent's window ออกมา

ส่วนอื่น ๆ ที่ดีขึ้น

ในเรื่องของไฟล์เมมโมในโปรแกรมประเภท dBASE-like ทั้งหลายจะมีข้อบกพร่องอันหนึ่งที่คล้าย ๆ กันก็คือ เมื่อเราลบเรคอร์ดใด ๆ ไปแล้ว ข้อมูลที่อยู่ในเมมโมไฟล์ไม่ได้ถูกลบไปได้ แม้ว่าเราจะใช้คำสั่ง PACK แล้วก็ตามข้อมูลในไฟล์เมมโมก็ยังคงอยู่ ถ้าหากเราจะลบข้อมูลในไฟล์เมมโมจริง ๆ ก็ต้องอาศัยคำสั่ง COPY TO (ดังที่แสดงไว้ในรูปที่ 7) ในฟอกซ์โปร

```
PROCEDURE PACKMEMO
PARAMETERS fname
SET EXCLUSIVE ON
USE (fname)
PACK
COPY TO temp.$$$
USE
ERASE (fname)
RENAME temp.$$$ TO (fname)
SET EXCLUSIVE OFF
RETURN
```

รูปที่ 7 การลบข้อมูลในไฟล์เมมโม

เอง แต่ก่อนก็มีข้อบกพร่องอันนี้อยู่เหมือนกัน แต่ทว่าในฟอกซ์โปร 2.0 แล้วจะมีคำสั่ง PACK MEMO เพื่อใช้ในการลบข้อมูลในเมมโมไฟล์ ซึ่งหลังจากที่ฟอกซ์โปร 2.0 เสร็จสมบูรณ์แล้วส่วนนี้ก็อาจจะมีการเปลี่ยนแปลงไปบ้าง

ผู้ที่เขียนโปรแกรมโดยใช้ภาษาในตระกูลดีเบส คงจะคุ้นเคยกับการใช้ตัวแปรมาโครกันพอสมควรและคงพอจะทราบว่า การใช้มาโครเนี่ยทำให้โปรแกรมช้าลงไปเยอะทีเดียว (ดังนั้นถ้าไม่จำเป็นก็ไม่ควรจะใช้พร่ำเพรื่อ) ใน dBASE IV และ FoxPro 1.x ได้มีส่วนที่ใช้อ้างถึงชื่อไฟล์แบบทางอ้อม (Indirect file reference) ที่สามารถใช้แทนมาโครได้ในการอ้างถึงชื่อไฟล์ ซึ่งให้ประสิทธิภาพสูงกว่าการใช้มาโคร สำหรับฟอกซ์โปร 2.0 แล้วได้ขยายเรื่องการอ้างถึงแบบอ้อมนี้ไปยัง การอ้างถึงชื่อฟิลด์, ชื่อเมนู, ชื่อวินโดว์ แต่ทว่าการอ้างถึงชื่อแบบทางอ้อมนี้จะใช้ไม่ได้ในคำสั่งบางคำสั่งเช่น LIST, DISPLAY, CALL

โดยส่วนตัว

สำหรับผู้เขียนเองได้ใช้ผลิตภัณฑ์ที่เป็น DBMP ในการเขียนโปรแกรมและพัฒนาระบบงานมาตั้งแต่การเริ่มต้นอาชีพโปรแกรมเมอร์ ความจริงแล้วในตอนนั้นผู้เขียนก็มีความต้องการที่จะใช้ภาษามาตรฐาน ในการเขียนโปรแกรมและพัฒนาระบบงาน เนื่องจากมีข้อดีกว่าการใช้ DBMP อยู่หลายประการ และเรื่องที่เป็นจุดเด่นของภาษามาตรฐานก็คือ การเปิดกว้างสำหรับแนวความคิดในการเขียนโปรแกรม โดยโปรแกรมเมอร์สามารถสร้างสรรค์งานขึ้นได้ตามจินตนาการของตนเองโดยไม่มีขีดจำกัด (นอกจากขีดความสามารถของอุปกรณ์ฮาร์ดแวร์) แต่ทว่าถ้าหากเรามองดูอีกมุมบ้างก็จะพบข้อดีของการใช้ DBMP ในการพัฒนาระบบงานอยู่ก็มีมากมายเช่นกัน และจุดดีที่เด่นชัดก็เห็นจะเป็นความสะดวกในการเขียนโปรแกรม เพราะว่าภาษาของ DBMP ได้จัดเตรียมฟังก์ชันต่าง ๆ ไว้ให้เรียบร้อยแล้ว โดยเฉพาะเรื่องของการจัดการฐานข้อมูล ซึ่งจะช่วยให้สามารถพัฒนาระบบงานได้อย่างรวดเร็ว ซึ่งส่งผลให้ประหยัดต้นทุนในเรื่องค่าใช้จ่าย ซึ่งเราต้องคำนึงถึงเรื่องนี้ไว้ให้มาก เพราะเรากำลังจำเป็นต้องแข่งขันกับทั่วโลกในเรื่องของซอฟต์แวร์ในเวลาอีกไม่นานนัก (ถ้ากฎหมายคุ้มครองทรัพย์สินทางปัญญา มีผลใช้ในเมืองไทย) เพราะถ้าหากเราใช้ภาษามาตรฐานในการเขียนโปรแกรม เราก็ต้องรับภาระเรื่องของการสร้างหรือจัดหา รูทีน สำหรับการจัดการฐานข้อมูลเอง และยังมีเหตุผลอีกหลาย ๆ อย่าง (ที่ผู้เขียนไม่ได้กล่าวถึง) เสริมให้ DBMP โดดเด่นขึ้นไปอีก โดยเฉพาะเมื่อพูดถึงความสามารถของ DBMP ในยุคปัจจุบันนี้แล้ว แทบจะกล่าวได้ว่าไม่มีการปิดกั้นแนวทางและรูปแบบในการพัฒนาระบบงานเหลืออยู่เลย ในอนาคตเราอาจจะสามารถเขียนโปรแกรมเกี่ยวกับกราฟิก, CAD, งานควบคุมเครื่องจักร ได้อย่างมีประสิทธิภาพด้วยภาษาของ DBMP ตัวใดตัวหนึ่งก็ได้ ใครจะไปรู้...